H.264 QoS and Application Performance with Different Streaming Protocols

Sanna Laine
University of Jyväskylä
P.O. Box 567
FIN-67701, Kokkola, Finland
sanna.laine@chydenius.fi

Ismo Hakala
University of Jyväskylä
P.O. Box 567
FIN-67701, Kokkola, Finland
ismo.hakala@chydenius.fi

ABSTRACT
Streaming techniques, including the selected streaming protocol, have an effect on the streaming quality. In this study, the performance of three different streaming protocols in a disturbed communication channel is evaluated with a modified version of the FFPlay player. A H.264 encoded video is used as a test sequence. The number of displayed image frames, the frame rate and playout duration are used as objective metrics for QoS. The metrics brings out differences of streaming protocols in our test environment. They are measured at the application level and have a connection to the user experience.

Categories and Subject Descriptors
C.2.2 [**Computer – Communication Networks**]: Network Protocols – *applications*; C.4 **[Performance of Systems]** – *performance attributes*; H.5.1 **[Information Interfaces and Presentation]**: Multimedia Information Systems – *video*.

General Terms
Measurement, Performance, Experimentation.

Keywords
QoS, QoE, H.264, streaming protocols, RTSP, RTMP, HLS.

1. INTRODUCTION
In an unimpaired video streaming session, the video plays smoothly, maintaining the frame rate, and displayed image frames are decoded correctly. Interferences in data transmission may degrade streaming quality. Cabled networks are more resistant than wireless networks to signal interference, but both transmission ways may suffer from congestion. Congestion may cause packet loss, decreased throughput, delay and jitter. Decreased network QoS affects the QoS of the application. Typical streaming client's reactions caused by insufficient network conditions are e.g. grown initial buffering time, rebuffering, blockiness, frame freezing, jerkiness and frame dropping. The user observes stalling, motion discontinuities and distorted video images, which all decrease quality of experience (QoE).

In this paper, we study if different streaming protocols cause different player performance when H.264 video is transmitted through a disturbed communication channel. The FFPlay player is used in the simulations. The H.264/AVC standard includes several techniques which have enabled lower bit rate encoding that maintains the same quality compared to its predecessor in the MPEG group. In addition to coding efficiency, its development has put emphasis on the adaptability to various networks and error resiliency. To handle a variety of applications and networks, the H.264/AVC design covers a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). VCL is designed to represent the video content efficiently compressed [21], [12]. NAL formats the VCL presentation of the video to make it more adaptable to a variety of transport layers or storage media [21], [12]. The error resiliency schemes in H.264/AVC are mainly contained in the VCL [10].

Effective compression usually means weaker error resiliency. The compression schemes aim to reduce redundancies from the data. On the other hand, the error resiliency schemes utilize redundancy in the data [9]. The decoder behavior in the presence of errors has been described in the standard, but the error concealment is not within the scope of the H.264/AVC standard [9].

Regardless of the additional attention to network friendliness, the H.264/AVC may have some weaknesses compared to e.g. MPEG-4 Part 2 codec. Van der Auwera *et al.* [18] found that the H.264/AVC codec achieves the lower average bit rates at the expense of significantly increased traffic variability. Their streaming simulation studies over a bottleneck link showed that the increased bit rate variability results in significantly higher frame losses for H.264/AVC encoded video compared to MPEG-4 Part 2 encoded video when transmitting a single video stream. Even when reducing the traffic variability with smoothing before transmitting, the smoothed H.264/AVC video traffic exhibits variabilities at the same level or above the unsmoothed MPEG-4 Part 2 video traffic.

The streaming protocols can be grouped into push- and pull-based protocols. In push-based protocols, the server streams packets to the client until the client stops or interrupts the session. In pull-based streaming, the client is active and requests content from the media server [1]. The advantages of pull-based protocols are simplicity and robustness, and when the server bandwidth is above several times of the streaming rate, the pull-based protocol is optimal in terms of bandwidth utilization and system throughput in P2P streaming [17]. The streaming protocols used in our simulations are Real-time Streaming Protocol (RTSP), HTTP Live Streaming (HLS) and Real Time Message Protocol (RTMP). RTSP is one of the most common session control protocols used in push-based streaming (although sometimes considered pull-based since the client needs to initiate the

MOBIMEDIA 2015, May 25-27, Chengdu, People's Republic of China

DOI 10.4108/icst.mobimedia.2015.259061

session). RTSP uses Real-time Transport Protocol (RTP) for data transmission. In addition to UDP, RTP may also be used with other suitable underlying network or transport protocols like TCP.

Apple's HLS is a pull-based adaptive streaming protocol which stores the video in multiple files called chunks. Client fetches video segments from the server, using the HTTP GET method. A new video segment is only requested after the previous one has been fully received. Segments are transported by TCP and are 10 seconds long by default.

RTMP is a TCP-based stateful, pull-based streaming protocol developed by Adobe Systems. It was designed to stream audio, video and data between Flash platforms. RTMP transmits information in *Message*s which are split into *Chunks*. There are about a dozen types of Messages. The payload of a Message is cut into the same size blocks of data, and the Chunk Header is added in front of each data block. In addition to the plain RTMP protocol, it has multiple variations. RTMPT is encapsulated within HTTP request. Although tunneling helps traversing firewalls, it causes latency [5]. RTMPS is RTMP over a TLS/SSL connection. RTMPE is RTMP encrypted using Adobe's own security mechanism.

The underlying transport protocol may have a major impact on the streaming quality. In our tests, all streaming protocols were sent via TCP and RTP packets, in RTSP sessions also via UDP. Hoßfeld *et al*. [8] studied the influence of the transport protocol on QoE with Youtube. They concluded that for the same bottleneck capacity the end user will tolerate stalling effects that arise in case of TCP better than temporal video artifacts that manifest themselves in case of UDP.

The remainder of this paper is organized as follows: in Section II, the related work is briefly discussed. Section III describes our test environment. In Section IV, the player's behavior with different protocols is observed in terms of frame rate monitoring. In Section V, the number of displayed image frames is compared between protocols in poor network conditions. The results of simulations are discussed in Section VI. Finally, we conclude the paper in Section VII.

2. RELATED WORK

Various studies have omitted the strategy to explore the streaming quality evaluation by collecting parameters at the application level. Dalal *et al.* [2]-[4] studied RTSP streaming (RTP over UDP) in Windows environment to develop methods for assessing user-perceived quality with objective metrics obtained from an instrumented media player. In those studies, the player's statistics about packet traffic, throughput and buffering are monitored to predict QoE. Packet loss and delay was induced, and they tried to select a loss rate that would achieve the desired amount of interference visible to user. Application level metrics in Windows environment are also used for measuring streaming quality e.g. in [19] and [14].

Wang *et al.* [20] collected streaming data with RealTracer, a tool collection that measures the performance of RealNetworks Video. The streaming protocol was RTSP. RealTracer includes RealTracker, a customized player that records system performance statistics and user perceptual quality ratings. RealTracker gathers information of a variety of metrics, but their study focused on frame rate, jitter, bandwidth and users' quality ratings.

Mok *et al.* [11] characterized the correlation between the application and network QoS with HTTP streaming protocol. For HTTP streaming, they chose buffering as the best metric for quality evaluation. They identified the rebuffering frequency to be the main factor responsible for the MOS variance.

French *et al.* [7] implemented a modified Flash player which collects QoS and QoE data during RTMP streaming. They customized their stream quality assessment system, originally developed for RTSP/UDP streaming, to make it capable to infer QoE of RTMP video streams. Their preliminary study indicated that bitrate in combination with either frame rate or bandwidth serves as an accurate indicator of QoE for RTMP videos.

In this study, different streaming protocols are compared in the same environment to bring out their possible differences in maintaining the streaming quality in poor network conditions. The number of displayed image frames, the frame rate and playout duration are measured at the application level. These metrics have a connection also to the user experience.

3. TEST ENVIRONMENT

FFplay is a simple and portable media player using the FFmpeg libraries and the SDL library. It is open source and supports multiple streaming protocols [6]. A simple player may bring out, more explicitly, the impacts of network impairments. FFmpeg, including FFplay, was installed on a client PC running Ubuntu 14.04. The source code of the player was slightly modified to be more suitable for the streaming quality monitoring of application layer. The pixel information and the display time of video pictures were collected. This enabled saving video pictures as they were displayed after real time decoding and concealing, including all possible errors, for prospective PSNR analysis. With the help of the collected frame display, time changes in frame rates could be observed. FFplay includes packet queues for audio, video and subtitles. The player fills those queues with data read from its buffers. For RTSP protocol, the size of queues is not limited as default. For HLS and RTMP, FFplay only fills the packet queues to a point, where they contain a specified amount of data. The maximum size of limited queues is 15*1024*1024 bytes. During the same test condition setting, the queue size was set equal for all protocols.

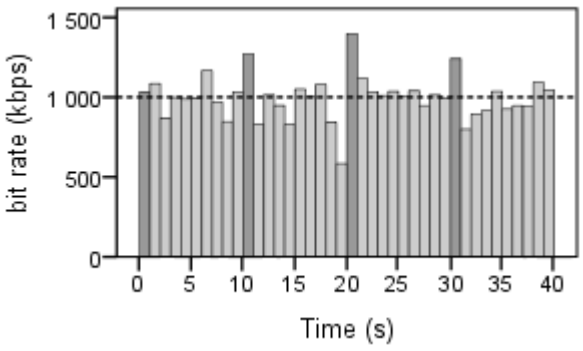

Figure 1. The bitrate of the test sequence. Bars including a key frame are darker.

The test sequence was created by concatenating four commonly used test videos: "Pedestrian area" (pa), "Sunflower" (sf), "Rush hour" (rh) and "Station" (st). The sequences were originally downloaded from LIVE Video Quality Database [15], [16]. At the database, those videos are downsampled to the spatial resolution

of 768x432 pixels and they are approximately 10 seconds long. The concatenated 40 second h.264 video was encoded with FFmpeg using x264 encoder. The target bitrate was set to 1000 kbps and the frame rate to 25 fps. The key frame interval was 250 frames, and the four key frames were in the beginning of each concatenated sequence (scene). The video includes no audio. The default profile, High@L3.0, of FFmpeg encoding was used. Figure 1 represents the bit rate profile of the sequence.

The test sequence was streamed as on-demand from a Wowza Streaming Engine 4.0.3. Limited and lossy communication channel was emulated with a Linktropy 5500 WAN emulator. The emulation was performed only from server to client to let feedback from the receiver travel unimpaired. The server and the client were on their own subnet and other network traffic was minimized. The server was connected to the emulator through a fibre channel. The adaptive bitrate video streaming wasn't used to simulate the playout behavior at the lowest bitrate with each protocol. With TCP-based protocols, the TCP variant was Cubic. The testbed is presented in Figure 2.

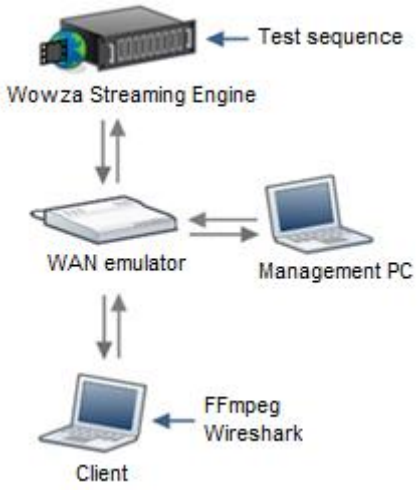

Figure 2. The test environment.

4. FRAME RATE DEPICTING USER EXPERIENCE

Frame rate monitoring during streaming depicts well the progress of the playout and brings out the changes that user perceives in streaming quality. In a network containing packet loss or insufficient throughput, a player may decrease the playout rate or start rebuffering. In those cases, the player may still display all the frames included in the video, which increases the playout time. The player may drop image frames, which also appears to the user as sporadic fluidity breaks. Breaks can be hardly noticeable, or they can be longer frame-freezing periods. They are often caused by the decoder's strategy to discard the video frame that is corrupted and repeat the previous frame instead, until the next valid decoded frame is available [13]. All those reactions by the player cause changes in the frame rate.

As network conditions were weakened in our environment, FFPlay reacted primarily by dropping image frames with TCP-based streaming protocols. The displayed image frames were flawless in most cases. The viewer observes the frame drops as sporadic breaks or jerkiness. At the packet loss ratio of 8%, the player drops video images with every tested streaming protocol in the test environment. Figure 3 shows examples of frame rates when packet loss is set to 8% and thus indicates the differences in the behaviors of protocols with FFplay player. The three highest diagrams (UDP/RTSP, TCP/RTSP and HLS) illustrate well the jerkiness a viewer observes. With the RTMP protocol, the playout stops once and the video picture freezes for a few seconds. Other than that, the sequence plays smoothly. However, in the playout with UDP/RTSP, the displayed image frames were also severely distorted.

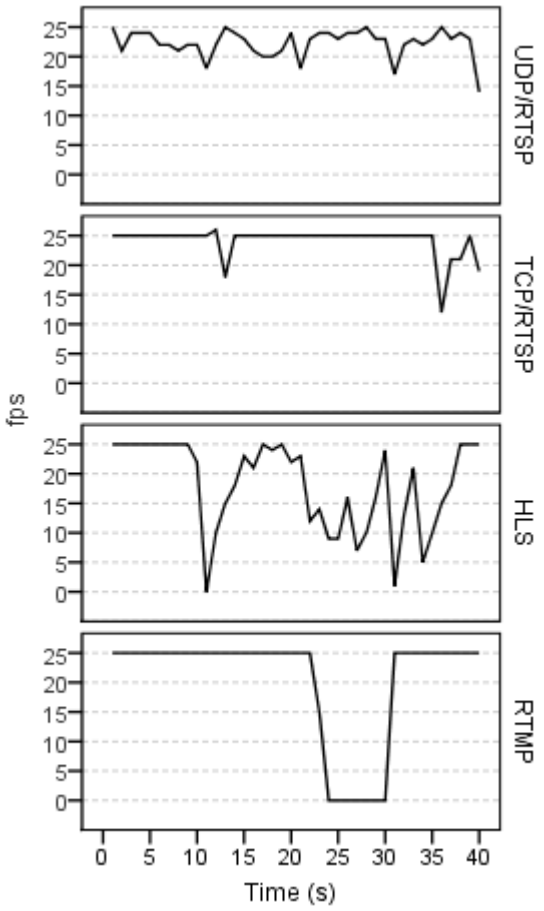

Figure 3. Examples of frame rates when packet loss ratio is set to 8% and the size of video queue is limited.

The bandwidth limit which introduced visible changes in the streaming playout with every tested protocol was 900 kbps in our testbed. In Figure 4, examples of frame rates are shown at the bandwidth of 900 kbps. The figure demonstrates how RTMP manages to keep the frame rate longest. With HLS, the playout time increases. This indicates that the player uses either very short rebuffering periods or decreases playout rate to cope with insufficient bandwidth. In these examples, RTMP displays most frames, 968 out of 1000. HLS displays 953, RTSP via UDP 873 and via TCP 785 frames.

The irregularities in playout often occurred near the key frames. The key frames lay in our test sequence only in the scene cuts. In addition, the observed quality could vary significantly between scenes. Figure 5 demonstrates the differences in scenes during the playout. The data was collected from the same test runs as in Figure 4. The crosses connected with a dashed line depict the time spent on each scene compared to the actual duration. The dots

connected with a solid line are the portions of displayed video pictures. If the streaming conditions were sufficient, both values would stay at 100%. In RTSP/TCP streaming, the third concatenated clip ("rush hour") loses most frames. The streaming gets better at the final scene, managing to play almost all the frames, but the increased time spent on it indicates unsmooth playout. Playing the 40 second test sequence took approx. 44 seconds with HLS. With RTMP, the first three scenes were played perfectly.

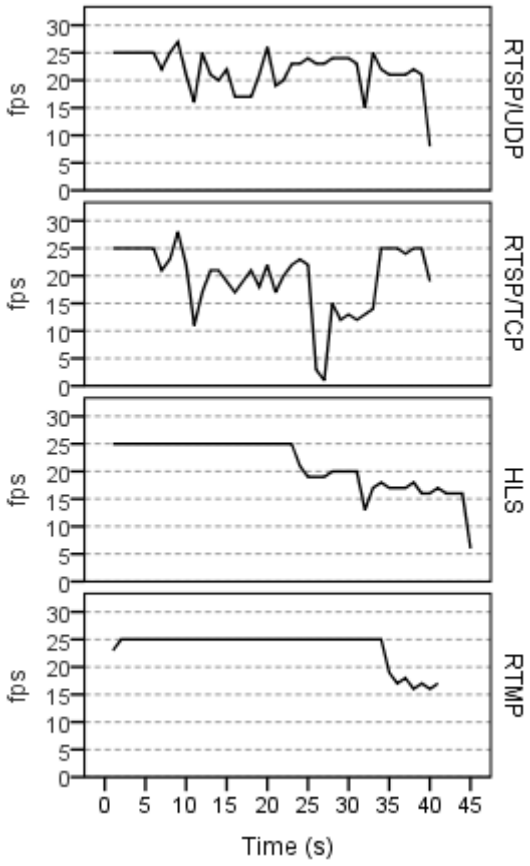

Figure 4. Examples of frame rates when the bandwidth is limited to 900 kbps. The video queue size was not limited.

5. RESULTS

The streaming protocol effects on application performance in poor network conditions were studied by inducing packet loss, and limiting bandwidth with the WAN emulator. To simulate the player buffer size modification, the tests were conducted with two different player's video queue sizes. Reducing the queue size better brings out defects in streaming and may highlight the differences between the streaming protocols.

As the FFPlay reacted primarily by dropping image frames, the ratio of displayed video frames was measured for each streaming session. The metric is related to changes in the frame rate. When image frame dropping was present, there was always frame rate variation. In this chapter, the bandwidth and packet loss ratio limits for the quality changes demonstrated in the previous chapter are sought for.

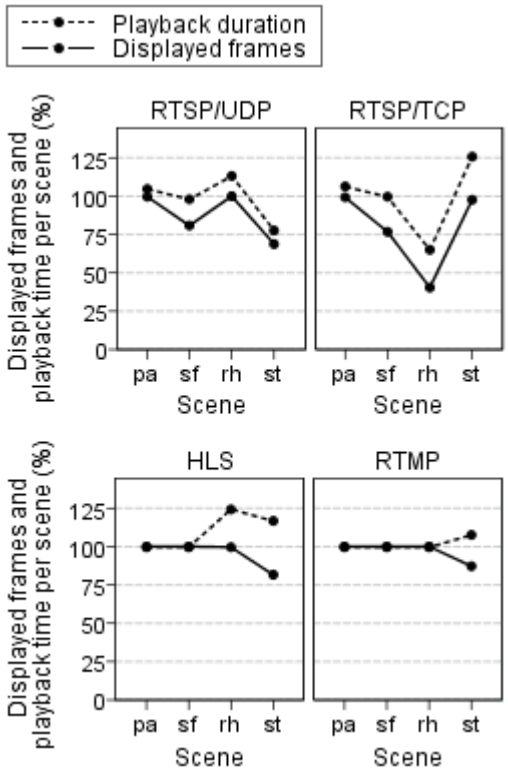

Figure 5. Examples of displayed video pictures and the time spent on playout when bandwidth is limited to 900 kbps. The video queue size was not limited.

5.1 Packet loss

The WAN emulator discards packets randomly, based on the specified loss rate. Since packets are dropped after they have been rate throttled, discarded packets will consume link bandwidth [10]. Figure 6 represents the number of displayed video pictures when packet loss was increased from 1 to 10% at the bandwidth of 2Mbps. The test video was played 5 times with each setting. In Figures 6 (a) – (d), the video queue size of FFplay was not limited. With the RTSP/TCP protocol, the first interferences were observed when the loss was 4%. The defects increased at a steady rate as the loss percentage was raised. For HLS, the limit was 8% and, for RTMP, 9%. For these settings, RTMP was the only one for which the connection broke in the middle of streaming, but before that it was able to maintain flawless playout. According to the figure, in RTSP/UDP streaming the number of displayed image frames and packet loss rate are directly proportional. Nevertheless, RTSP/UDP couldn't cope even with 0.5% loss ratio but displayed severely distorted video pictures. Clearly, FFplay doesn't provide any support for UDP resending.

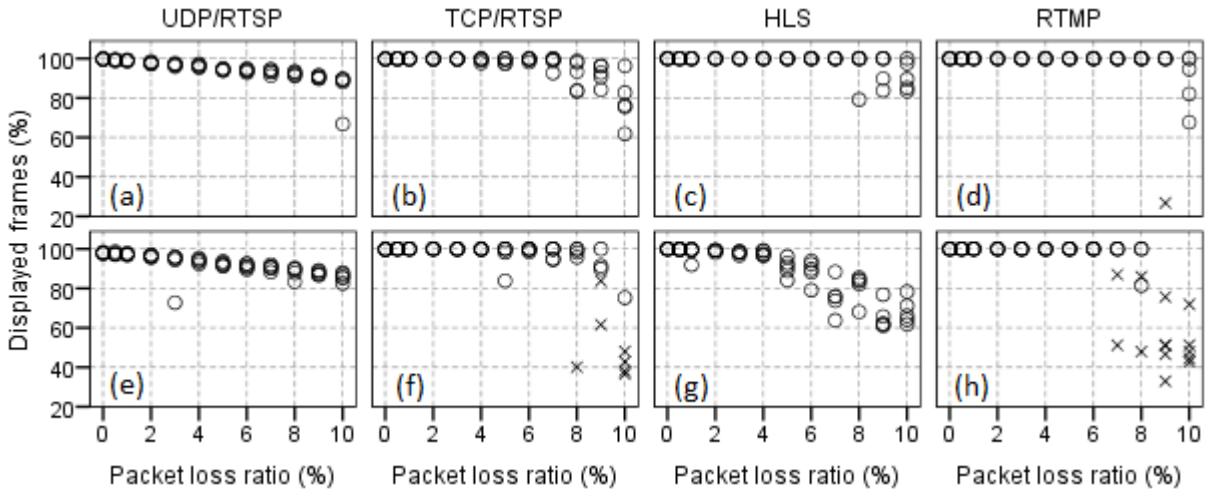

Figure 6. Displayed video pictures as a portion of all image frames in a test sequence in lossy channel. The value marked with cross depicts a test case when streaming connection broke before the end. In (a) – (d), the video queue size was not limited, whereas in (e) – (h), the size was limited.

In Figures 6 (e) – (h), the size of the player's video queue is limited. Since the quality of RTSP/UDP streaming was already poor, the change in the queue size didn't make any fundamental difference in quality. HLS couldn't display all frames from 1% and RTSP/TCP from 5% onward. RTMP again maintained perfect quality until 7%. HLS may lose more frames, but it manages to keep the connection and play the whole sequence with these settings. Because the type of the lost packet has a great impact on propagation of errors and maintaining the streaming quality, there is some deviation also in the displayed image frames metric.

5.2 Bandwidth Limitation

The WAN emulator throttles the frames to the specified WAN bandwidth. Frames in excess of the specified WAN bandwidth are queued to the configured emulator's maximum queue depth. When the queue is full, newly-arriving frames are discarded [10]. To explore player's reactions in slow bandwidth condition, large queue depth, 10 000 ms, was used to minimize packet loss caused by the emulator. Bandwidth was limited from 2 Mbps downward. The portion of displayed frames with both video queue sizes is presented in Figure 7. In tests with restricted bandwidth, the sequence was played almost identically every time at the same setting. As in packet loss, the frame dropping manifested itself as jerkiness and short interruptions. With RTSP/UDP, corrupted video pictures also occurred.

In Figures 7 (a) – (d), the video queue size was not limited. The streaming was fluent with all protocols until the bandwidth of 1 Mbps, which was expected as the average bitrate of the test sequence was about that size. Below 1 Mbps, HLS started to drop some video pictures, causing jerkiness. RTMP got to the point it couldn't maintain the connection to the server. RTSP streaming reacted by dropping video pictures more radically, and lost also the connection a few times. When RTP packets were streamed on top of UDP, there were also distorted pictures at the bandwidth of 800kbps. All those distorted pictures featured at the final concatenated 10 second sequence in the 40 second video.

In Figures 7 (e) – (h), the video queue size was limited. With RTSP/UDP, the player displayed severely distorted video pictures already at the 2 Mbps bandwidth. As the bandwidth was decreased near 1 Mbps, video pictures were displayed perfectly, with just minor jerkiness. The transient quality improvement may indicate that, with faster bandwidth, player's video queue fills up and discards the excess packets. When the bandwidth was limited to 800 kbps, the video pictures in the final 10 seconds were distorted as in the case where video queue wasn't restricted.

The change in the video queue size didn't cause any fundamental differences in the player's performance with RTSP/TCP. In HLS streaming, the jerkiness started at higher bandwidth than with the unlimited queue size. Similarly, in RTMP streaming, the connection was lost already at the bandwidth of 900 kbps.

6. DISCUSSION

In this study, different streaming protocols were compared in the same environment in a communication channel including packet loss and unsufficient bandwidth. For these network parameters, we tried to find boundaries after which the streaming quality starts to decrease. According to the simulations, there are clear differences in streaming quality between TCP- and UDP-based streaming protocols. The difference became emphasized due to the fact that there was no application layer resend mechanism implemented for UDP. This means, that every dropped packet will cause artifacts that the user perceives, and the number of dropped packets may work alone as a good quality indicator. Usually the resend mechanism is implemented, and, in that case, for example the number of retransmitted packets should be used.

For TCP-based protocols, the frame rate, the ratio of dropped/displayed picture frames and the playout duration monitoring depict well the progression of streaming playout and stalling effects. These metrics can be measured from the application layer and changes in them can be assumed to be perceptible to the user. Especially the frame rate changes bring out the characteristics of the streaming protocols. The ratio of dropped/displayed picture frames indicates changes in frame rate and/or duration of playout. Thus, the number of dropped/displayed picture frames is a metric that compresses the information and is more suitable for evaluating large samples.

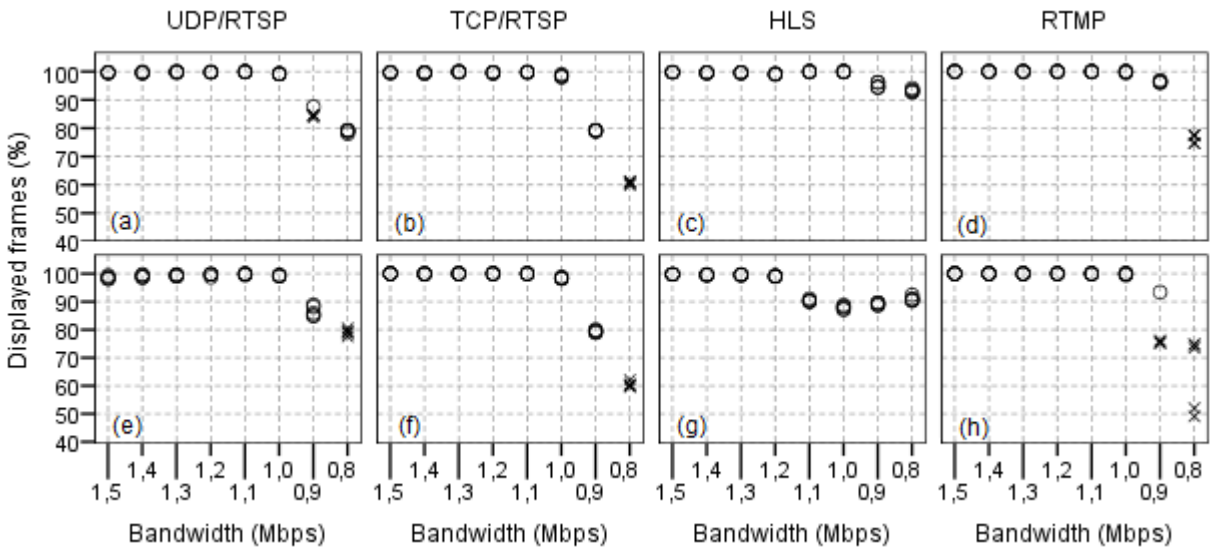

Figure 7. Displayed video pictures as a portion of all frames in a test sequence with limited bandwidth. The values marked with cross depict test cases when streaming connection broke before the end. In (a) – (d), the video queue size was not limited, whereas, in (e) – (h), the size was limited.

There were two pull-based protocols in our tests. The HLS protocol was the only one which didn't break the connection during the poor network conditions. On the other hand, when the player's video queue size was reduced, the streaming via HLS protocol was first to show changes in quality in both lossy and limited bandwidth conditions. This may indicate that the protocol consumes more bandwidth than the other TCP-based protocols. RTMP managed to keep the quality slightly longer than the others.

In addition to missing UDP resend support, the FFPlay also defines different video queue sizes to RTSP when compared to HLS and RTMP as defaults. In these simulations, this was taken into account and altered. However, this demonstrates that players compatible with multiple streaming protocols may contain implementations that aren't equal.

7. CONCLUSION

In this study, the effects of the RTSP/RTP, HLS and RTMP protocols on player's reaction in disturbed network conditions were observed. The study shows that, when delivering streaming media in a platform compatible with multiple streaming protocols, the quality may differ in same network conditions between streaming protocols.

RTMP seemed to keep the quality slightly longer than the others. HLS was the most resilient not breaking the connection once during our tests. RTSP/TCP may induce more variation in quality when packet loss is present. On the other hand, when bandwidth was limited, reducing player's queue size didn't have any effect on the quality. RTSP/UDP performed worst due to the lack of UDP resend mechanism.

To expand the research, more developed player supporting multiple streaming protocols should be studied. With the simple media player used, the concealing methods aren't very sophisticated.

8. ACKNOWLEDGMENTS

The authors wish to thank the European Regional Development Fund for the grant to the "Media Centre Lime – the Regional Innovation and Knowledge Environment" project, the European Social Fund for the grant to the "SmartCampus" project and the Executive Agencies, for their help and all the partners of the project for their contribution.

9. REFERENCES

[1] Begen, A. C., Akgul, T., and Baugher, M. 2011. Watching video over the web, part I: Streaming protocols. *IEEE Internet Comput.* 15, 2 (March/April 2011), 54–63. DOI= http://dx.doi.org/10.1109/MIC.2010.155.

[2] Dalal, A. C., Musicant, D. R., Olson, J., McMenamy, B. , Benzaid, S., Kazez, B., and Bolan, E. 2007. Predicting user-perceived quality ratings from streaming media data. In *Proceedings of the IEEE International Conference on Communications* (Glasgow, Scotland, June 24-28, 2007). ICC '07. IEEE, 65–72. DOI= http://dx.doi.org/10.1109/ICC.2007.20.

[3] Dalal, A. C., and Purrington, K. 2005. Discerning user-perceived media stream quality through application-layer measurements. In *Proceedings of the First International Conference on Multimedia Services Access Networks* (Orlando, FL, June 13-15, 2005). MSAN '05. IEEE, 44–48. DOI= http://dx.doi.org/10.1109/MSAN.2005.1489940.

[4] Dalal, A. C., and Perry, E. 2003. A new architecture for measuring and assessing streaming media quality. In *Proceedings of the third Workshop on Passive and Active Measurements* (San Diego, CA, April, 2003).

[5] DeRienzo, F. 2013. Tunneling with RTMP encapsulated in HTTP (RTMPT) should be avoided as it causes latency, http://blogs.adobe.com/connectsupport/tunneling-with-rtmp-

encapsulated-in-http-rtmpt-should-be-avoided-as-it-causes-latency/, Adobe Systems Inc., November 2013.

[6] FFPlay Documentation, December 2014. https://www.ffmpeg.org/ffplay.html.

[7] French, H., Lin, J., Phan, T., and Dalal, A. C. 2011. Real time video QoE analysis of RTMP streams. In *Proceedings of the 30th IEEE International Performance Computing and Communications Conference* (Orlando, FL, November 17-19, 2011). IPCCC'11. IEEE, 1–2. DOI= http://dx.doi.org/10.1109/PCCC.2011.6108105.

[8] Hoßfeld, T., Schatz R., and Krieger, U. R. 2014. QoE of YouTube video streaming for current Internet transport protocols, measurement, modelling, and evaluation of computing systems and dependability and fault tolerance. *Lect. Notes in Comput. Sc.* 8376 (2014), 136–150.

[9] Kumar, S., Xu, L., Mandal, M. K., and Panchanathan, S. 2006. Error resiliency schemes in H.264/AVC standard. *J. Vis. Commun. Image R.* 17, 2 (April 2006), 425–450. DOI= http://dx.doi.org/10.1016/j.jvcir.2005.04.006.

[10] Linktropy WAN emulator user's guide, Firmware Version 4.4, Apposite Technologies, May 2014.

[11] Mok, R. K. P., Chan E. W. W., and Chang, R. K. C. 2011. Measuring the quality of experience of HTTP video streaming. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management* (IM) (Dublin, Ireland, May 23-27, 2011). IEEE, 485–492. DOI= http://dx.doi.org/10.1109/INM.2011.5990550.

[12] Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., and Wedi, T. 2004. Video coding with H.264/AVC: tools, performance, and complexity. *Ciruits and Systems Magazine, IEEE* 4, 1 (2004), 7-28. DOI= http://dx.doi.org/10.1109/MCAS.2004.1286980.

[13] Quan, H., and Ghanbari, M. 2009. No-reference temporal quality metric for video impaired by frame freezing artefacts. In *Proceedings of the 16th IEEE International Conference on Image Processing* (Cairo, Egypt, November 07-10, 2009). ICIP'09. IEEE, 2221–2224. DOI= http://dx.doi.org/10.1109/ICIP.2009.5413894.

[14] Reibman, A.R., Subhabrata, S., and Van der Merwe, J. 2004. Network monitoring for video quality over IP. In *Proceedings of the Picture Coding Symposium* (San Francisco, CA, December 2004).

[15] Seshadrinathan, K., Soundararajan, R., Bovik, A. C., and Cormack, L. K. 2010. Study of subjective and objective quality assessment of video, *IEEE T. Image Process* 19, 6 (June 2010), 1427–1441. DOI= http://dx.doi.org/10.1109/TIP.2010.2042111.

[16] Seshadrinathan, K., Soundararajan, R., Bovik, A. C., and Cormack, L. K. 2010. A subjective study to evaluate video quality assessment algorithms. In *Proceedings of Human Vision and Electronic Imaging*, 7527 (January 2010).

[17] Zhang, M., Zhang, Q., Sun, L., and Yang, S. 2007. Understanding the power of pull-based streaming protocol: Can we do better?. *IEEE J. Sel. Area. Comm*., 25, 9 (December 2007), 1678–1694. DOI= http://dx.doi.org/10.1109/JSAC.2007.071207.

[18] Van der Auwera , G., David Prasanth T., and Reisslein M. 2008. Traffic and quality characterization of single-layer video streams encoded with the H.264/MPEG-4 advanced video coding standard and scalable video coding extension, *IEEE T. Broadcast*., 54, 3 (September, 2008), 698–718. DOI= http://dx.doi.org/10.1109/TBC.2008.2000422.

[19] Wang, Z., Banerjee, S., and Jamin S. 2003. Studying streaming video quality: from an application point of view," In *Proceedings of the eleventh ACM international conference on Multimedia* (2003). ACM, New York, NY, 327–330. DOI= http://dx.doi.org/10.1145/957013.957083.

[20] Wang, Y., and Claypool, M. 2005. RealTracer - tools for measuring the performance of RealVideo on the internet. *Multimedia Tools and Applications* 27, 3 (December 2005), 411-430. DOI= http://dx.doi.org/10.1007/s11042-005-3757-6.

[21] Wiegang, T., Sullivan, G. J., Bjøntegaard, G., and Luthra, A. 2003. Overview of the H.264/AVC video coding standard. *IEEE T. Circ. Syst. Vid.* 13, 7, (July 2003), 560–576. DOI= http://dx.doi.org/10.1109/TCSVT.2003.815165.